

**Amendments to the Specification:**

**Please replace paragraph 0013 with the following replacement paragraph:**

[0013] The proliferation of test results and the corresponding test analysis resources have not been met with sufficient technological advancements in reducing the labor involved in test result analysis. Currently, test result files can be differentiated, whereby identical result files can be categorized together. This provides some help in allowing test result analyzers to group identical failures over multiple lab runs, but result files may differ slightly even if ~~[[the]]~~ a failure occurred for the same reason, simply because the failure occurred in different computing environments. Categorization based on entire result files therefore often requires redundant attention from result analyzers to slightly different result files.

**Please replace paragraph 0041 with the following replacement paragraph:**

[0041] Parsing a result file refers to breaking it into smaller chunks so a program can act upon the information. In this regard, imagine an XML result file 200 such as that of Fig. 2. ~~The illustration of Fig. 2 is highly simplified and illustrates a result file 200 with far less information than a typical test result file, but it is instructive in illustrating parsing.~~ A first line in the ~~input~~ result file 200 identifies a test that was conducted, "open a file." A second through sixth line identifies ~~some~~ scenario information for the test, ~~and so on~~. Parsing such a file 200 can involve writing a program such as driver 201 that extracts these lines. An ~~input~~ A test result file 200 can be in any file format and can be consistently formatted according to a particular document structure to facilitate operations on the parsed result files. To use XML parlance, ~~an input~~ a result file 200 can conform to any schema. Schemas ensure that test data in a result file 200 are consistently tagged and structured, so that parsing operations can be easily performed

**Please replace paragraph 0042 with the following replacement paragraph:**

[0042] The actual properties, or failure data, that is extracted by the driver 201 can vary at the discretion of those skilled in the art. Failure data should include relevant test result information that is likely to be useful for analysis.~~[[.]]~~ In the context of software testing, there are several result file 200 properties that are often considered useful in

analyzing test failures. The following are examples of these properties not intended to limit the invention, but rather to demonstrate potential properties to be extracted from a result file 200. First, it may be desirable to extract the actual output of a tested operation. In other words, if a tested operation is “add the number four and the number five,” and the expected result was “nine” but the actual output was “seven,” it can be useful to have this information for analysis. Second, a test result itself is certainly a likely candidate for extraction from a test result file 200. A test can be passed or failed, or some other result that provides additional information about the result may be returned, such as “warning” indicating that the tested operation returned a warning, “exception,” indicating that the tested operation returned an exception, “unknown” indicating that the result of the tested operation is not known, “timeout,” indicating that the operation did not complete, or “aborted,” indicating that an operation was aborted. Third, call-stack information, indicating the calls that were made in the course of the tested operation, may be useful to extract from a test result file 200. Fourth, any exceptions that were generated in the course of the tested operation may be useful to extract from a test result file 200.[[.]]